Adaptive Color Classification for Structured Light Systems

Philipp Fechteler and Peter Eisert
Fraunhofer Institute for Telecommunications - Heinrich-Hertz-Institute
Image Processing Department,
Einsteinufer 37, D-10587 Berlin, Germany
{philipp.fechteler / peter.eisert}@hhi.fraunhofer.de

Abstract

We present a system to capture high accuracy 3D models of faces by taking just one photo without the need of specialized hardware, just a consumer grade digital camera and beamer. The proposed 3D face scanner utilizes structured light techniques: A colored pattern is projected into the face of interest while a photo is taken. Then, the 3D geometry is calculated based on the distortions of the pattern detected in the face. This is performed by triangulating the pattern found in the captured image with the projected one.

The main focus of our work lies in the enhancement of the systems robustness with respect to environment illumination, color cross-talk, reflectance characteristics of the scanned face etc. For this purpose the color classification of the proposed system is made adaptive to the characteristics of the captured image to compensate for such distortions. Further improvements are concerned with enhancing the quality of the resultant 3D models. Therefore we replace the typical general-purpose image preprocessing with specialized low-level algorithms performing on raw CCD sensor data.

The presented system is suitable for generating high speed scans of moving objects because it relies only on one captured image. Furthermore, due to the adaptive nature of the used color classifier, it generates high quality 3D models even under perturbing light conditions.

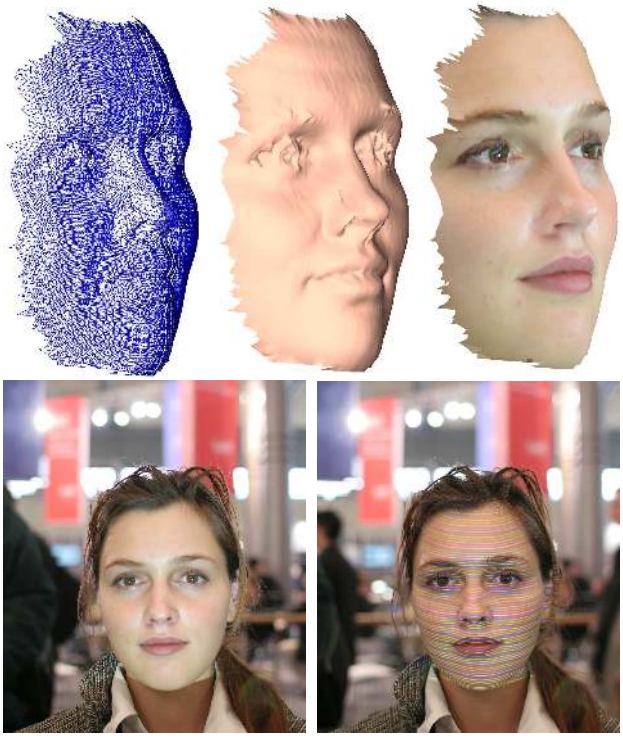

Figure 1. **top:** Resulting 3D models as wire frame model, surface and textured surface, **down:** input images

1. Introduction

The construction of 3D models out of 2D views on a scene is a field of ongoing research for some decades now. One common way of approaching this problem is Stereo Vision. In this case the corresponding points of two or more different views are triangulated to create a 3D model. A good overview and evaluation on such algorithms is given in [6].

Very similar to the Stereo Vision approach is the Structured Light method which is used in this work. Here the task is simplified by using controlled illumination. There are various different structured light approaches, for example: in [12] a real-time system is proposed which runs on specialized hardware; in [7] a method is presented for generating high resolution depth maps of complex scenes by using multiple projectors, cameras and several snapshots per camera; in [9] a method is shown which uses just one projector and one camera without any modifications running on a typical PC. This last mentioned work has motivated us to develop a structured light 3D scanner specialized for faces [2], which poses the foundation of the presented sys-

tem.

In recent research significant effort has been made to enhance the performance of such systems with respect to the resulting 3D models. In [10] high quality depth maps are generated by a spacetime stereo technique which is based on a video stream captured while the projected pattern changes. In [11] the authors present an approach to capture high resolution 3D models of faces utilizing several synchronized video cameras.

The major contribution of the presented work lies in the improvement of structured light systems with respect to robustness to ambient light and reflectance characteristics of the object to be scanned. Additionally we present low level image processing algorithms suited for the generation of high accuracy 3D models.

2. Framework and Architecture

A 3D model of a face is computed by first projecting a simple colored stripe pattern onto the face. The depth information is then calculated by taking into account the distortion of the stripes in the face caused by its shape. To measure the degree of distortion, correspondences between projected and detected stripes are established. The depth is evaluated for all correspondences with respect to the focal point of the camera. After having a cloud of 3D points it is converted into a mesh of triangles. This mesh constitutes the surface of the 3D model. Optionally the mesh can be textured with a picture taken additionally with regular white light.

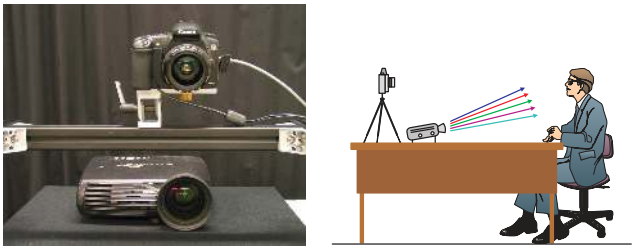

Figure 2. Devices and setting used in this framework

The hardware used by our framework consists of regular devices: a digital camera and a projector (see figure 2). Both devices are controlled by a typical PC running the framework. The devices are mounted so that their image centers are one upon the other.

In order to generate a 3D model of a face the following steps are performed:

1. take image I_{input} of the face illuminated with a color stripe pattern $I_{pattern}$ and optionally capture an image $I_{regular}$ with regular white light
2. extract prospective stripes
3. extract colors corresponding to prospective stripes
4. match the prospective stripes with the projected ones
5. calculate 3D coordinates of correspondences
6. create a triangle mesh from the 3D point cloud
7. optionally project $I_{regular}$ onto the surface as texture

To create a 3D model of a face, the goal is to find the most probable correspondences between $I_{pattern}$ and I_{input} among all possibilities. This is achieved by performing a global optimization after having extracted hypothetical stripes in I_{input} as well as their colors.

2.1. Offline Pattern Creation

The pattern projected onto faces should allow an easy assignment of imaged parts to parts of the pattern. Therefore we have chosen a stripe pattern with horizontal lines having fully saturated colors with empty (black) spaces in between. This reduces the search for correspondences to a 1D search along the corresponding scan columns. The colors in the resulting pattern image $I_{pattern}$ are (see figure 3): red, green, blue, white, cyan, magenta and yellow. To ease the unique assignment of detected stripes to projected ones we have chosen a series of stripe colors with a big period. Besides that, we introduced the constraint that two consecutive stripes have to differ in at least two channels. With this latter constraint we achieve an enhanced delimitation of successive stripes, and the unique identification is simplified. This is the reason why we do not use de Bruijn sequences [3], which are often used in similar contexts to generate sequences with big periods. The smaller periodicity due to the additional constraint in our case is no problem, as long as the period of the pattern is smaller than the largest jump in depth. Taking this into account the pattern $I_{pattern}$ can be determined with a simple depth-first search.

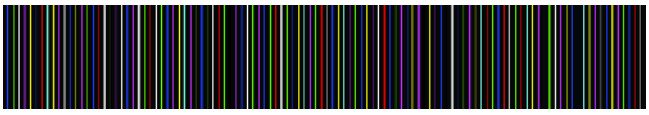

Figure 3. A cut-out of the used pattern rotated by 90°

3. Detection of Stripes

After capturing an image I_{input} of a face illuminated with the pattern $I_{pattern}$ the stripes corresponding to the projected pattern stripes are detected in I_{input}. First of all, the region of interest in the image is defined with a simple face shaped model. All the pixels outside are set to black, so that all subsequent steps will ignore them. The remaining image of the face will be searched for the projected stripes.

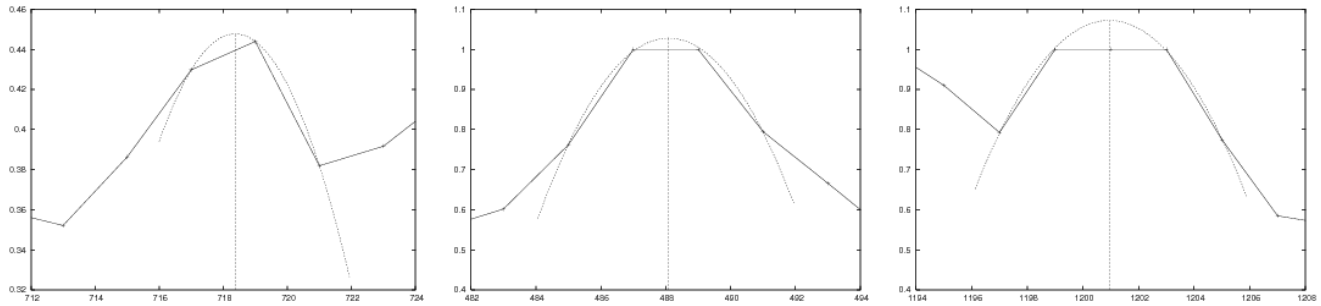

Figure 4. Different cases of parabola fitting through pixel values along scan columns; vertical line indicates maximum of parabola

3.1. Sub-Pixel Resolution

To achieve a highly accurate 3D model the stripes are detected with sub-pixel resolution.

Therefore all the "general-purpose" image preprocessing in the camera is skipped, which is generally "special-purpose" to generate visually appealing images: Bayer interpolation, gamma correction, white balancing etc. Instead, the presented framework uses the plain CCD sensor values with full 12-bit resolution instead of the typical 8 bits. Among others, this means, we treat the pixel values differently depending on which sensor type they were measured on, red, green or blue.

As the input image is in raw CCD sensor format, there are no de-bayered RGB pixels, but columns of single-channel pixels with alternating color sensitivity: RGRGRG... and GBGBGB...

Extracting the prospective stripes is done separately for all three color channels as well as for every scan column. A pixel is taken as a stripe candidate if the values of the preceding and succeeding pixels are not bigger. This results in three lists of stripe candidates, one for each sensor color type.

We are interested in sub-pixel resolution. So we determine the centers of the stripe candidates by fitting parabolas through their intensities, the pixel values: $p(x) = ax^2 + bx + c$, with a, b, c being the parameters of the parabola $p(\cdot)$ and x the pixel location along the scan column, again for all three color channels separately. The center of the stripes is assumed to be at the maximum point of the parabola, its mode. Fitting the parabolas is performed via squared distance minimization: $\min_{\mathbf{p}}(\mathbf{X} \cdot \mathbf{p} - \mathbf{y})^2$, with the parameter vector $\mathbf{p} = (a, b, c)^T$, the matrix $\mathbf{X}$ holding the different powers of the pixels locations along the scan column, and $\mathbf{y}$ containing the actual pixel values.

In the regular case with one pixel value bigger than its two vertical neighbours, the parabola is fitted through these three points. In cases where two adjoining pixels hold the same value bigger than the two surrounding ones, these four pixels are used for this. If there are more than two equal valued pixels, the inner most pixels are ignored. In this case the parabola is fitted with respect to the two starting and the two closing pixels of that interval. The latter case occurs when the sensor is saturated, e.g. too much light. See figure 4 for illustration of all these cases.

Most projected stripes produce responses in more than one kind of color sensor, e.g. magenta light should excite the sensors for red and blue. And even if a sensor gets illuminated with light it is not sensitive to, a response is measurable with probability bigger than zero. This is formally known as color-cross talk. That's why one projected stripe often results in several detected stripes which are found in the lists corresponding to the different sensor types.

In order to get one common list with all stripes found in the input image I_{input} the three lists are fused to one single set. Thereby detected stripes of different lists (emerging from different sensor types) which belong to one projected stripe are combined to one common representation. The common center c_{common} is calculated as a weighted sum of the two original centers c_1 and c_2:

$$c_{common} = \frac{c_1 a_1 + c_2 a_2}{a_1 + a_2},$$

weighted with their "sharpness" parameter in the parabola equation $p(x) = ax^2 + bx + c$. Here again a scan column is

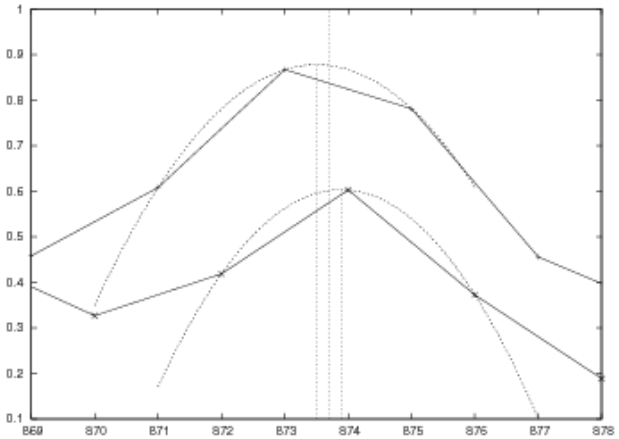

Figure 5. Two parabolas resulting in one common center

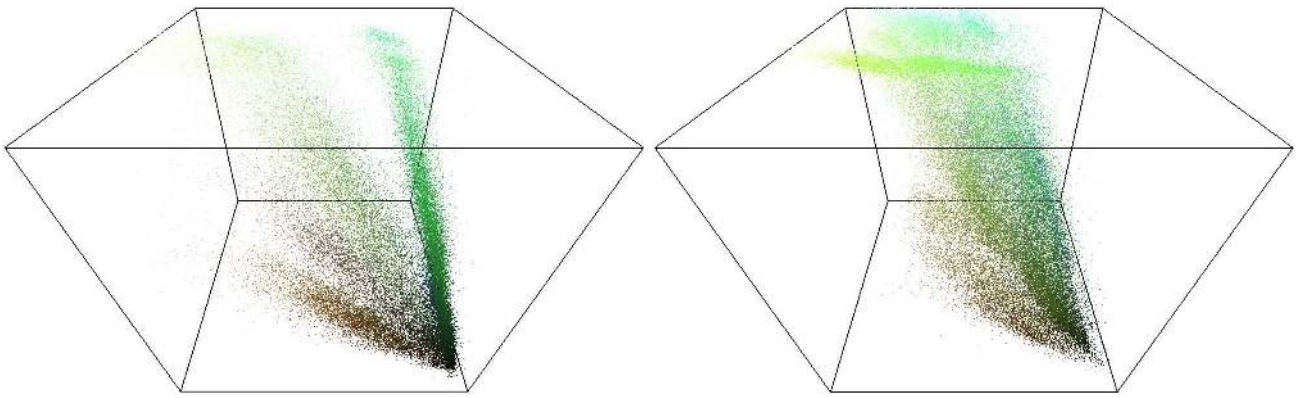

Figure 6. Colors of prospected stripes in RGB space, **left:** picture taken under ideal conditions, **right:** picture taken under bad illumination

processed one after another, but now under consideration of the two different color channels present. Every two parabolas in a single scan column from the two different sensor types are fused together if their centers are not to far apart, see figure 5.

For establishing correspondences the colors are compared between the projected stripes and the detected ones. Therefore every prospective stripe is a color assigned using sub-pixel Bayer interpolation. This means, that stripe candidates get their red, green and blue values by assigning them as a weighed mean of their neighboring sensor values. The weights used here are inverted Euclidean distances between the stripes' centers and the pixel locations.

3.2. Probability of Stripes

At the later over-all-optimization, detected stripes which deviate from the pattern sequence order to much and which are not bold enough are cancelled out. For this purpose every detected prospective stripe is assigned a likelihood of being a correctly recovered projected stripe. This likelihood is proportional to the same "sharpness" parameter of the fitted parabola as the one mentioned above.

All the parabolas are opened to the lower side, so $a < 0$. And parabolas with a low absolute value of a near zero are flatter then the ones with more negative values of a. To normalize the derived likelihoods $p_{i,valid}$ of a stripe candidate $\mathbf{p}_i$ to be a valid stripe to a range of $[0, 1]$ the sharpness parameter a_i is divided by the lowest value of all a's among all parabolas of the same sensor type.

$$p_{i,valid} = \frac{a_i}{a_{min}}$$

This means, that indistinct stripes will correspond to flatter fitted parabolas which in turn results in lower probability weights.

4. Color Classification of Detected Stripes

The result of the previous steps is a list of all the stripe candidates. Each stripe candidate is specified by a scan column index, a position along that scan column and a RGB color value. For every detected stripe the likelihoods of being projected with the different colors of the pattern are derived. Hence every stripes pixel is assigned one probability value for each projected color, seven in our case.

4.1. Classifying the Detected Colors

Experiments have shown that projected colors, reflected by skin and recorded by cameras encounter various different distortions. Additionally, sensor noise as well as color-cross talk is detected between the projector spectra and the sensor filters. In figure 6 two RGB space representations of prospective stripes are depicted: one captured under controlled conditions, the other one in a usual office environment. Color clusters are roughly identifiable corresponding to the projected colors, without a clear separation between them. The visible clusters are approximately shaped along more or less straight lines which seem to be slightly displaced versions of the black→red, black→cyan etc. axes. The plots in figure 6 show how crucial the light conditions of the environment are. Without any disturbing light sources in the environment, the clusters are identifiable quite clearly. But with increasing ambient light the clusters become more and more fuzzy until there is only one big blob of data points in RGB space.

4.2. Adaptation to Statistics in Captured Image

In order to determine the color each detected stripe pixel was projected with, straight lines $g_c : \mathbf{o}_c + x\mathbf{r}_c$ are fitted through these clusters to form prototypes of these clusters; one line for each pattern color $c \in \{r, g, b, c, y, m, w\}$. The classification of data point $\mathbf{p}_i$ (a stripes' color) is then performed by calculating the distances $d(\mathbf{p}_i, g_c)$ of that point

$\mathbf{p}_i$ to all the prototype lines g_c, and assigning the color of the prototype with the smallest distance.

This fitting of straight 3D lines through clusters is a form of orthogonal distance regression (ODR), and the classification of the projected colors is a form of model selection. The parameters for this mixture model (the straight lines g_c) are determined out of the measured data. For this purpose the KMeans algorithm [5] is adapted. The classical KMeans method works broadly in the following way:

1. initialize the parameters of the classifier, the mixture of straight 3D lines
2. repeat until no changes in labeling are registered
   * **label** the data with current classifier parameters
   * **adapt** classifier parameters

There are efficient general-purpose initialization methods for the standard KMeans method, e.g. [1]. These methods are not applicable for our adapted KMeansLineFit because here the cluster means are not in the same space as the data, but in parameter space of straight lines. However, by knowing the originally projected colors, our initial guess of classifier parameters are straight lines, originating from black (0, 0, 0) and pointing to the fully saturated colors red (1, 0, 0), magenta (1, 0, 1) etc.

The labeling step means to assign every stripe the color label it was most probably projected with according to the current classifier. This is the color of the prototype line g_c with the smallest distance to the stripes color $\mathbf{p}_i$.

The adaptation step is slightly more complex: Every prototype line g_c is moved into the center of the data points which are currently labeled with the same color. At a first glance this can be done for every cluster independently. The calculation of a straight line g_c in 3D space is performed by searching the offset point $\mathbf{o}_c$ and a direction vector $\mathbf{r}_c$. The parameters minimizing the squared distance to the given data are the datas' mean as the offset point $\mathbf{o}_c$, and $\mathbf{r}_c$ can be determined by using Eigenvalue decomposition of the datas' covariance matrix.

Experiments have shown that the resulting lines do not lie in the cluster centers, because all the clusters are fused together at dark colors. The prototype lines do not pass from dark colors near black to lighter colors near the fully saturated ones and they do not cross the clusters in their center. To overcome this, an artificial constraint has been introduced, that all the prototype lines must contain one common point near black, which is also adapted by the KMeansLineFit algorithm. This seems to be plausible as it is the pixels' value achieved when dimming any color more and more.

The problem of finding all prototype lines passing through one common point poses a system of coupled equations which are not solvable in closed form. Therefore it is solved approximately by first calculating the direction vector $\mathbf{r}_c$ for every color c, and then determining the common offset point $\mathbf{o}$ best fitting all the new prototypes g_c to the corresponding data. The former is again the same Eigenvalue decomposition problem. The later problem can be solved by minimizing the sum of squared distances for each given data point $\mathbf{p}_i$ and its corresponding prototype line g_c:

$$\min D = \sum_{c=1}^{C} \sum_{i=1}^{N_c} \mathrm{d}^2(\mathbf{p}_{ci}, g_c) = \sum_{c=1}^{C} \sum_{i=1}^{N_c} \|\mathbf{r}_c \times (\mathbf{p}_{ci} - \mathbf{o})\|^2$$

The offset $\mathbf{o}$ minimizing this squared distance D can be calculated by deriving D with respect to $\mathbf{o}$'s components and setting them to zero. The resulting three equations, resolved for $\mathbf{o}$'s components can be combined into an equation system of the form $\mathbf{A} \cdot \mathbf{o} = \mathbf{b}$ which can be solved with stable matrix inversion.

After having iterated over the labeling-adaptation-loop until no changes in labeling are registered, the classifier is adapted to the statistical characteristics of the input image. For every stripe the distance to the different prototype lines is defined. Figure 7 shows the results of an example.

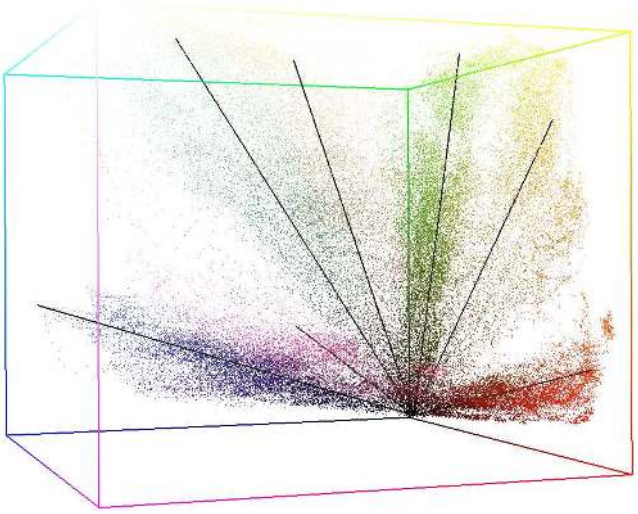

Figure 7. Prototype lines found by adapted KMeansLineFit in RGB space

4.3. Probability of Color Assignments

These distance measurements are transformed into likelihoods by utilizing a softmax like function. The distance measurements are inverted and normalized to the range of $[0..1]$ with the sum of the reciprocal of all distances:

$$p_{i,color} = \frac{d^{-1}(\mathbf{p}_{i,color}, g_{color})}{\sum_{c=1}^{C} d^{-1}(\mathbf{p}_{i,c}, g_c)}$$

Special care has to be taken for RGB pixel values close to $\mathbf{o}$ to ensure that no division by zero occurs.

To speed up the convergence an intelligent initialization can be used: Fit an origin passing line through the complete data set and let it be the white representative. The remaining prototypes are put aside the white prototype into the direction of the fully saturated axes, depending on the expansion of the whole data point cloud.

With the presented method we have a soft color classifier which assigns probabilities in contrast to absolute values. Additionally by utilizing the proposed non-parametric KMeansLineFit method the soft color classifier is adapted to the characteristics of the given input image I_{input} in terms of color-cross talk, albedo etc. without explicitly modeling these effects.

5. Matching Detected Stripes with Projected Ones

During the previous steps a list has been achieved containing all the detected stripe candidates. Each one is specified by a location (scan column index and position along that one), a likelihood to be a valid stripe and the likelihoods to be projected with the pattern colors. The current task is to establish correspondences between projected and detected stripes and to skip all invalid stripe candidates which have been emerged due to non-optimal light, skin and sensor conditions. This constitutes the probabilistic over-all-optimization.

This matching is a typical combinatorial optimization problem (COP): Which combination of correspondences fits best. We follow the usual way to solve such tasks by setting up an objective function which has to be maximized in order to find the best combination. The objective function takes all the available information for all stripe candidates into account, which is:

- likelihood to be a valid stripe ($p_{i,valid}$)
- likelihood to be projected with the different pattern colors ($p_{i,color}$)
- deviation of detected sequence from projected pattern ($p_{i,sequence}$)

This problem is solved for each vertical scan column separately. The objective function we have developed is the sum over all the available probabilistic weights. We distinguish between the two cases of stripe candidates are matched ($\mathbf{p} \in M$) and skipped ($\mathbf{p} \notin M$):

$$\begin{aligned} L = & \sum_{\forall i \in M} \quad p_{i,color} + p_{i,valid} + p_{i,sequence} \\ + & \sum_{\forall i \notin M} \quad 1 - p_{i,valid} \end{aligned}$$

L contains the probability of being invalid for every rejected stripe candidate. For every successfully matched stripe it contains the sum of the likelihood of being a valid stripe, the likelihood that this stripe was projected with the corresponding pattern color and the likelihood that this color occurs in this sequence in the projected pattern. The latter term is often called a jump weight because it assigns good scores for stripes being in order with the pattern and bad scores for incoherent sequences.

This COP is solved efficiently with the Dynamic Programming (DP) method [8]. The typical Dynamic Programming approach is to set up a table containing scores for the assignments and traversing through it. Afterwards the best score achieved is traced back and all the encountered correspondences are found respectively the prospective stripes marked as invalid are skipped.

6. Experimental results

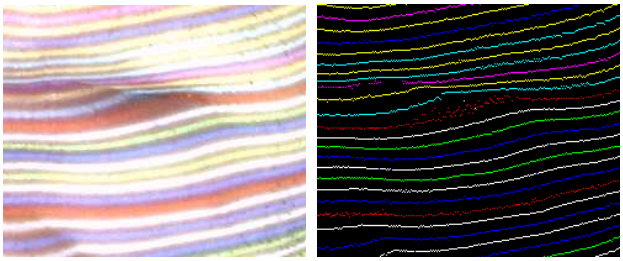

Figure 8. Results of stripe detection, color classification and stripe matching without any post-processing

The final depth of every correspondence is evaluated by triangulating the 3D point cloud. Therefore the projection matrices of the camera and projector are needed, which we get by calibration.

The depth calculation is done by calculating the intersection of the two lines of sight through the focal points and the image points of the camera respective projector [4].

In our experiments we use a DLP projector "Projection Design F1+" with a SXGA+ resolution (1400 x 1050) and a camera "Canon EOS 20D" with a resolution of 8.2 mega pixel (3522 x 2348). The pattern contains stripes with a width of 2 pixels and 3 pixels intersection.

The C++ running time for generating a 3D model of a face lies in the range of a minute on a 3 GHz Pentium-4 computer. It depends on the amount of detected vertices. The KMeansLineFit algorithm converges in 5 to 15 iterations for optimal light conditioned scenes. With captured images of non-ideal scenes the amount of required iterations rises up to 50.

Many experiments have been performed with the 3D face scanner. Figure 1 shows the results of a typical scenario. Two pictures have been taken, one with regular white light and one with the structured light pattern shown in figure 3. After selecting the interesting region, the system has set up a color classifier suitable for the given scene (shown in figure 7). After classifying the detected stripes and establish-

ing correspondences the 3D model of the face is calculated and optionally presented as wire frame model, surface or textured 3D face.

In figure 8 a region around the mouth is depicted to demonstrate the main stripe matching including stripe detection and color classification. Here no post processing has been performed which is normally done to remove resp. align outliers. Despite the bad quality and the partial saturated color channels in the input image the system produces good results.

7. Conclusion and Future Work

We have presented a system for high resolution 3D face scanning based on single captured images. The system generates high accuracy 3D models by exploiting specialized low-level algorithms performing on raw CCD sensor data. Additionally the 3D face scanner has been made robust in terms of light conditions, skin, color-cross talk etc. This is achieved by adapting the color classification to the characteristics of the captured image utilizing the proposed non-parametric KMeansLineFit algorithm without the need to explicitly model any of these disturbing effects.

Experiments with scanned faces under non-ideal light conditions are presented to demonstrate the systems performance.

The simple setup and its easy usage make the presented system ideal suited for various 3D model creation scenarios, e.g. virtual environments like 3D games or human machine interfaces.

Interesting for future work could be to use a global optimization for the whole input image, which does not handle every scan column separately. E.g. Dynamic Programming could match all scan columns simultaneously by taking all the possible combinations of current stripe matches as a single state. Another interesting enhancement could be to adapt the color classifier to local regions instead of the whole image. Additionally a significant speed up could be gained by running several parts of the system in parallel exploiting todays multi-core processors.

8. Acknowledgement

The work presented in this paper has been developed with the support of the European Network of Excellence VISNET II (Contract IST-1-038398).

References

[1] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Symposium on Discrete Algorithms, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027 – 1035, New Orleans, Louisiana, USA, 2007.

[2] P. Fechteler, P. Eisert, and J. Rurainsky. Fast and High Resolution 3D Face Scanning. In *Proc. of the International Conference on Image Processing ICIP*, San Antonio, Texas, USA, September 2007.

[3] H. Fredricksen. The lexicographically least debruijn cycle. *Journal of Combinatorial Theory*, 9:509–510, 1970.

[4] R. Goldman. *Intersection of Two Lines in Three Space*, page 304. Academic Press, 1990.

[5] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, 1967.

[6] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV 47(1/2/3)*, pages 7–42, April-June 2002.

[7] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 1, pages 195–202, Madison, WI, June 2003.

[8] D. B. Wagner. Dynamic programming. *The Mathematica Journal*, 1995.

[9] L. Zhang, B. Curless, and S. M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, Padova, Italy, June 2002.

[10] L. Zhang, B. Curless, and S. M. Seitz. Spacetime Stereo: Shape Recovery for Dynamic Scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, Wisconsin, June 2003.

[11] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. Spacetime Faces: High-Resolution Capture for Modeling and Animation. In *ACM Annual Conference on Computer Graphics (SIGGRAPH Proceedings)*, pages 548–558, Los Angeles, CA, August 2004.

[12] S. Zhang and P. S. Huang. High-resolution real-time 3-d shape measurement. *Optical Engineering*, 45(12), December 2006.